Co-simulation methods for holistic vehicle design: A comparison

2020-01-1017

Co-Simulation Methods for Holistic Vehicle Design: A Comparison

Nikolaos Kalantzis[1], Tom Fletcher[1], Ahmed Ahmedov[1], Ruoyang Yuan[1]
Antonios Pezouvanis[1], Kambiz Ebrahimi[1], Sina Shojeai[2], Richard Osborne[3]

[1]Loughborough University, UK
[2]Jaguar Land Rover, UK
[3]AVL, UK

Abstract

Vehicle development involves the design and integration of subsystems of different domains to meet performance, efficiency, and emissions targets set during the initial developmental stages. Before a physical prototype of a vehicle or vehicle powertrain is tested, engineers build and test virtual prototypes of the design(s) on multiple stages throughout the development cycle. In addition, controllers and physical prototypes of subsystems are tested under simulated signals before a physical prototype of the vehicle is available. Different departments within an automotive company tend to use different modelling and simulation tools specific to the needs of their specific engineering discipline. While this makes sense considering the development of the said system, subsystem, or component, modern holistic vehicle engineering requires the constituent parts to operate in synergy with one-another in order to ensure vehicle-level optimal performance. Due to the above, integrated simulation of the models developed in different environments is necessary. While a large volume of existing co-simulation related publications aimed towards engineering software developers, user-oriented publications on the characteristics of integration methods are very limited. This paper reviews the current trends in model integration methods applied within the automotive industry. The reviewed model integration methods are evaluated and compared with respect to an array of criteria such as required workflow, software requirements, numerical results, and simulation speed by means of setting up and carrying out simulations on a set of different model integration case studies. The results of this evaluation constitute a comparative analysis of the suitability of each integration method for different automotive design applications. This comparison is aimed towards the end-users of simulation tools, who in the process of setting up a holistic high-level vehicle model, may have to select the most suitable among an array of available model integration techniques, given the application and the set of selection criteria.

Introduction

Modern vehicles comprise of mechanical, electrical, and hydraulic systems, as well as hardware and software [1]. As a result of the gradual addition of functionality in automotive subsystems, the electrification of powertrains, and the introduction of autonomous driving the complexity of vehicles increases with time [2]. In turn, the above has increased the complexity of the vehicle development process [3], while at the same time, the number of offerings of a given vehicle model tends to increase [4] All the above lead to an increased design space and required volume of calibration data, and in combination with the tendency of vehicle development cycles tending to become shorter, and development budgets to be reduced [5] make Computer Aided Engineering an indispensable tool of automotive research and development.

CAE substitutes physical prototypes of an engineering system with its mathematical representation with the intent to produce data that are valid and applicable in the physical world [4]. Computer aided engineering (CAE) has revolutionized the way complicated engineering systems are developed. Instead of relying on expensive and time-consuming physical prototypes, engineers can now test a virtual prototype that predicts the behavior of the design before a physical prototype or the 3D geometry of the components is available, thus CAE is implemented from the early stages of automotive development [6]. As a result, a multitude of designs can be evaluated at a fraction of the time and cost required for a single prototype to be built and evaluated, and this leads to an increased design maturity and fitness for purpose at a lower cost as the number of required physical prototypes is considerably reduced. In addition, subsystems and components can be developed and tested before physical prototypes of neighboring components are available (frontloading) thus further shortening the development cycle [7]–[9]. Due to the above, CAE has a wide spectrum of applications [6] and is well-established and used throughout the development cycle of a road vehicle [10], with each automotive component or subsystem being developed by a specialized team of engineers [1] using application-specific CAE tools [4], [7], [10].Traditionally, each automotive subsystem and component has been developed and tested by a specialized team of engineers [1] using a separate development approach [9], [11]. Subsystem development has taken place in isolation from the rest of the system, and subsystem performances as parts of the whole system/vehicle have been tested once physical prototypes of all components/subsystems were assembled to a complete physical vehicle prototype [9]. A subsystem or component connected to and interacting with other subsystems or components may behave in a manner that is not observed during standalone simulation or physical testing and which leads to a reduced or even unsafe system level performance. While the traditional design approach also known as sequential system design [12] can yield highly successful product designs, it does not consider the effects component and subsystem interaction may have on their performance and the performance of the whole system [8] and therefore it does not facilitate the early-stage design defect detection, which in turn may lead to a design defect being carried undetected to later or even the final design stages [9]. Design defects detected at later developmental

stages tend to be difficult and expensive to correct [12] as a redesign from an earlier stage may be necessary [10]. In addition, a globally optimal system-level design necessitates that all subsystems are designed to operate in perfect synergy with one another [6], [9], [11]–[13]. From the above, one may realize the great importance of addressing the dependence of component/subsystem performance on their interaction with the whole system [9] which is reflected in the modern holistic/multi-disciplinary design approach.

In the modern holistic approach, a complete virtual prototype of the whole system is built and tested well before a physical prototype of the whole system is available [12] and each subsystem and its respective control algorithm are developed as parts of the whole system rather than in isolation [9]. This allows for the early detection of subsystem incompatibilities, and the design of components and subsystems that work in synergy, thus allowing for the generation of globally optimal system-level designs [6], [12]. In addition, this facilitates the early development of subsystems and/or controllers that would normally be developed at later stages (frontloading), and thus, the parallelization of the development process, and this leads to shorter, more efficient development cycles [1], [5], [6], [9], [14]. The holistic model must be capable of describing certain system dynamics in the detail required by the particular task/area of study [12]. Through the incorporation of the communication models within a holistic system model, the failure modes caused due to communication errors can be studied [7], thus making controller debugging and testing possible in early design stage, leading to faster and cheaper controller development [6], [13]. Through the coupling of a physical subsystem (or a group of) such as a powertrain component to real-time capable virtual prototypes of subsystems whose physical prototypes are not yet available (XiL), the physical subsystems can be verified early on in the vehicle development cycle [1]. Using a holistic virtual prototype can reduce risks of damage to expensive physical prototypes by reducing the required volume of physical prototype testing and in some cases by identifying hazardous test combinations on a virtual test bed and avoiding them on the physical testing [7]. Due to the above, the holistic design philosophy is used throughout the vehicle development cycle, from the early verification and validation of system requirements, to subsystem and component development, to XiL applications for embedded controller development and vehicle validation.

A numerical model of an engineering system is as good as the value (accurate results) it brings to the engineering toolbox for the respective effort (development and simulation costs). At low (component and subsystem) design level, it is in common practice for different departments within the vehicle development cycle, to use the best suited modelling and simulation environment for the given area of study [4], [7], [10], thus taking advantage of specialized component libraries, numerical solvers, workflows, and user interfaces to make the best numerical model for a given amount of effort [6]. Such practice leads to a collection of subsystem and component models built in a plethora of different modelling and simulation environments that are incompatible with one another. This diversity in used CAE tools across the vehicle development cycle facilitates the development of low-level subsystem high-performing models but constitutes a barrier to the integration of the low-level subsystem models into high-level systems, and thus is not conductive to the implementation of a holistic vehicle design approach as the integration of heterogenous software into a holistic model not only has to overcome the software incompatibility but also allow for the integrated models to operate under different time scales and numerical solvers [12], [13].Thus, model integration may be difficult and time consuming [10]. CAE software developers are addressing this problem by adding model integration capabilities to their products either in the form of model import or model export, and there are currently several methods available to choose from. Each method is characterized by different advantages and disadvantages and significant gains in workflow efficiency and quality can be made if the selected method is well suited to the application. The simulation engineer must select the most suitable model integration method for solving a given model integration problem, a task which by itself is not always straightforward, mainly due to the limited amount of available information on an end-user perspective. The aims of the current article are to review the current trends in model integration within the automotive sector, evaluate the interface performances under different use cases and criteria, and formulate end-user-oriented guidelines regarding the suitability of each evaluated interface for a given application.

Model Integration in Literature

A literature review has been carried out with the purpose of constructing a map of the uses of model integration within the automotive sector. The reviewed literature is categorized and briefly described in the following paragraphs.

Automotive Embedded Control Design

P. Le Marrec et al. [1] co-simulated software C Code, VHDL hardware model, and MATLAB mechanical component models using VCI integration interface with the purpose of carrying out the functional validation of the initial ECU specification.

Guoxing Li et al. [15] setup a co-simulation consisting of a CarSim vehicle dynamics model and a Simulink ABS controller model with the purpose of comparing a novel ABS control algorithm to a baseline ABS control algorithm.

F. Xie et al. [16] co-simulated an AMESim torque converter model with a MATLAB/Simulink transmission control unit and validated the co-model under a typical passenger car drive cycle

F. Renga et al. [11] developed a co-model between an injection control software model running in a PC, a controller hardware model running in FPGA, and Simulink based neural network model of the electromechanical parts with the purpose of designing the injection control.

M. Maharun et al. [17] built a high level model of a PHEV by co-simulating an ADAMS/Car vehicle model with a Simulink model containing the electrical components, the vehicle dynamics controller, and the energy management system and used the co-model to evaluate the performance of the vehicle dynamics controller and the energy management system in terms of handling characteristics and energy efficiency respectively.

Taotao Wu et al. [8] co-simulated a GT-Power engine model, an AMESim model of the torque converter the transmission and the vehicle dynamics, and a Simulink model of the engine controller and the shift controller serving as the global model with the purpose of investigating the potential coordinated engine and gearbox control has for improvement in vehicle fuel efficiency and shift quality.

Lars Mikelsons et al. [4] setup a co-model in Model.CONNECT between an FMU model of a yaw rate controller created in ETAS EVE, an FMU model of a vehicle dynamics created in CarMaker, and

an FMU model of the powertrain model created on GT-Suite, and used the co-simulation to carry out the functional validation of the yaw rate controller.

P. Casoli et al. [13] co-simulated a MATLAB S-Function fluid power model created in AMESim with a Simulink ICE model with the purpose of producing an optimal fluid circuit design and a fuel-efficient control strategy for mobile machinery.

S. Li et al. [18] co-simulated an ADAMS/CAR multibody dynamics vehicle model with a Simulink ESP controller model and tested the ESP control strategy performance.

Fuel Consumption Optimization

O. Özener et al. [19] used a proprietary integration interface to setup a co-simulation between an IPG Truck Maker 3D articulated bus and road model, and an AVL CRUISE drivetrain model in order to optimize the speed profile of city busses in terms of emissions and fuel consumption.

J. J. Eckert et al. [20] co-simulated a Simulink longitudinal dynamics model with an ADAMS multibody dynamics vehicle model and optimized the gear shifting strategy in terms if vehicle performance and fuel consumption.

System Design – Combination of Subsystems and System-Level Performance – NVH Studies

I. M. Khan et al. [6] integrated an FMU multibody dynamics 3D vehicle model originally built in ADAMS to an LMS AMESim driveline and powertrain controller model and used the co-simulation to predict vehicle noise, vibration, and harshness.

A. Karvonen et al. [2] co-simulated an ANSYS Simplorer electric machine and electric drive model with an ANSYS Maxwell magnetic component model of the electric machine with the purpose of studying the current and voltage harmonics induced by switching events on the DC bus of an electric drive.

Development of Automotive Test Rig and Test Scheduling

Serge Klein et al. [5] integrated a dSpace VSM vehicle dynamics model, a GT Power Fast Running Engine model, an FMU CS model of an automatic double-clutch transmission originally built in Simulation X, an ASM Tool Suite vehicle model, and a Simulink transmission control model. The Model in Loop (MiL) co-simulation was setup on dSpace VEOS and used to validate the concept of Engine in Loop (EiL), and then to commission a physical EiL test cell.

Tom Fletcher et al. [21] co-simulated a Ricardo WAVE-RT model of a GTDI engine as a local model of a Simulink based engine test cell controller and PCM global model by means of the dedicated WAVE-RT interface block. The co-model was used in the development of an automated engine calibration validation tool.

B. Zhang et al. [14] setup a co-simulation of a vehicle suspension durability test rig comprising of a mechanical components model in ADAMS, and a hydraulic and control elements model in Simulink. Co-simulation control was achieved via the use of a virtual server by Remote Parameter Control Pro Software.

The review of literature shows that model integration methods allow for the development and testing of software, hardware, and communication networks, using a high-fidelity model of the controlled plant, under conditions of normal but also faulty operation and for these reasons, they are indispensable in the development of reliable, high performing vehicle controllers. In addition, model integration allows for constructing highly detailed holistic vehicle models by connecting well-correlated, high-fidelity subsystem models developed by the most suitable discipline-specific environments into a vehicle-level simulation useful in identifying vehicle driving patterns and subsystem control strategies that give an optimum combination of fuel consumption and emissions, and performance. In addition, a detailed simulation of subsystem interaction is used to verify that the combined systems are compatible and to avoid combinations with a low performance. Other uses of co-simulation enable the safe and low-cost offline development of vehicle test rigs and the physical testing of components, subsystems, and even the complete vehicle under realistic and repeatable conditions. The FMI standard and the proprietary platform coupling interfaces are the most popular means of model integration. Simulink is the most popular co-simulation target environment hosting control algorithms for control development. AVL Model.CONNECT is also a popular environment dedicated to building heterogenous high-level models and carrying out holistic simulations.

Methods for Model Integration

The fitness of a model for a given purpose is directly related to the extent to which it encapsulates what is required by the application level of detail [9]. Model integration in general, and co-simulation in particular, allows for combining highly detailed models of different domains [4] built in a variety of area-specific modelling and simulation environments [5], [7], [8]. As a result, the strengths of the different environments are combined [6], [14], while at the same time, software heterogeneity associated obstacles are overcome [16], and therefore, a multidisciplinary/holistic design approach is made possible [1]. By using co-simulation, engineers incorporate high-level dynamic behavior into a numerical model, and develop and evaluate subsystems and control strategies under a multi-disciplinary/holistic approach [12]. Simulation speed of computationally expensive models can be accelerated to reach real-time via the export of the model(s) to FMI or an S-Function [4] and the simulation of the exported models on a real-time computer. In controller development applications, it is a common practice to use real-time co-simulation [3]. Real-time co-simulation is also an integral component of HiL testing which connects physical components/subsystems to a real-time simulation of other automotive components/subsystems [5]. Due to the advantages of the concept of model integration, its popularity is increasing within the automotive industry [3].

Most of the modern modelling and simulation environments support the connection to other modelling and simulation environments [8] via model import, model export, or a coupled simulation. There is an array of integration methods in use, and each method is characterized by a different combination of advantages and disadvantages. With respect to the universality of application, model integration interfaces can be classified into two main categories:

Proprietary interfaces. This category includes all interfaces that are proprietary to and connect only to a specific target environment. Such interfaces usually involve the two connected models running on their native environments and the two simulations exchanging data via a virtual network [8], [17], [20]. The majority of CAE software offer

proprietary interfaces to connect to Simulink as an imported or exported model.

Environment Agnostic Interfaces. Such interfaces do not have a specific target environment but are rather universal in nature as they are supported by a large percentage of commercial and open source CAE software for import and export. The most prolific example of an environment agnostic interface is the Functional Mockup Interface (FMI), which depending on version of the FMI standard, and the location of the numerical solver, is divided into different subcategories. Another interface that can be considered tool agnostic due to being used by both MATLAB/Simulink but also by real-time computers.

The tree diagram of the most popular model integration methods is shown in figure 1. On the top level, a distinction between Model Exchange (ME) and co-simulation is made. In model exchange, the model is exported by its native environment in a format that is readily importable by a third-party environment (such as FMU ME). The imported model is simulated within the third-party environment as part of the whole simulation using a single numerical solver available in the libraries of the third-party software. In co-simulation, each of the integrated models runs on its own native numerical solver, and the execution and communication of the multiple local simulations is coordinated by a global simulation. Co-simulation is divided into two solver configurations. In the standalone configuration, the exported model is packaged with its native numerical solver and when integrated, the local simulation of the imported model takes place in the third-party environment using the included native solver. In the platform coupling configuration, a co-simulation wrapper is exported to allow for a virtual network to be established between the native and the third-party environments. Each model is simulated on its original environment and the execution and communication of all local simulations are coordinated by the global simulation on the native or a third-party environment. The advantage of co-simulation over model exchange is that it is capable of multi-resolution simulations and allows for multiple domain-specific solvers to be integrated into one high-level multi-disciplinary simulation [22] that handles the stiffness of each model [13], as well as controller sampling rates. In co-simulation, the communication between the connected models takes place at every macro-step. Each of the local models is simulated under its own micro-step. In a hard coupling setup, the micro-step is equal to the macro-step, while in a weak coupling setup, the micro-step is smaller than the macro-step. A weak coupling setup relies on the local simulations extrapolating their inputs on micro-steps between macro-steps, a feature that introduces errors and inconsistencies in the calculations, and as a result, a weak coupling co-simulation is potentially less accurate than hard coupling co-simulation.

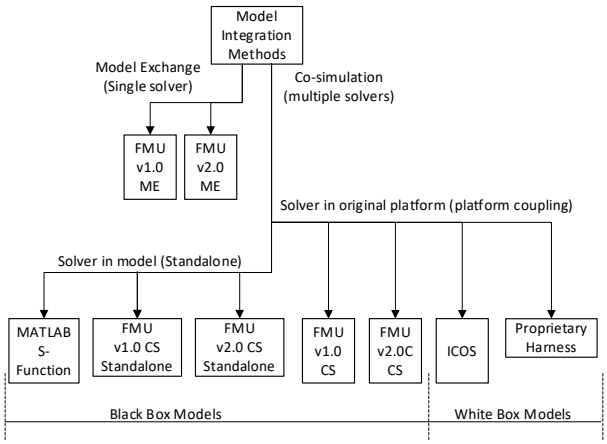

Figure 1. Tree diagram of model integration methods

Comparison of Model Integration Interfaces

As discussed above, there is an array of model integration interfaces and it is common for CAE tools to incorporate more than one within their libraries. The structural differences between the different types of interfaces translate to different sets of advantages and disadvantages which define the suitability of each interface for a given application. Thus, choosing the best interface for the given application or developmental stage can benefit the development process, and for this reason, this selection must be the product of a careful consideration.

The Functional Mockup Interface (FMI) standard describes universal model interface specifications and has been developed by a consortium of research groups and CAE tool developers [10]. Initially as a tool for design of embedded vehicular systems, it has become popular in other engineering sectors [7], [23], and is currently the most prolific and promising environment-agnostic model integration standard as it is supported for import and export in various forms by numerous CAE tools [6], [7], [10], [24]. The FMI-compliant model file is essentially an archive file and it is called a Functional Mockup Unit (FMU). Two versions of the FMI standard exist while FMI the release of version 3.0 has been announced. Compared to FMI 1.0, FMI 2.0 has some additional functionality. All FMI versions support Model Exchange (ME), Co-Simulation (CS), and Co-Simulation Standalone (CS – Standalone) solver configurations. In all co-simulation applications, data exchange is discrete point [10]. In the FMI CS configuration, the local and global models are simulated in their native environments and the communication between the environments is achieved via a virtual server, and therefore, it can be classified as a platform coupling method. The FMI CS – Standalone configuration involves the export of the model and the native solver to one FMI compliant file which is imported to a third-party environment and ran as a local simulation using its original solver. The FMI standard is currently the most important tool for CAE connectivity with a few shortcomings:

No global simulator is specified by the standard [7]

No high-level software approaches such as object-oriented development is included in the FMI specifications [10]

There is no support for vectors and structures, and as a result, there is no way to model the timing of the communication network between

the blocks without modelling a virtual CAN. This complicates ECU representation [4]

In the case of the FMU for Model Exchange, the model within a CAE tool is connected to an FMI I/O bus and the model is exported to FMU ME specifications and in turn, it is imported by a third-party CAE tool and connected to a third-party model. The imported model becomes an integral component of the third-party simulation as both the FMU ME and the third-party model are simulated by the same third-party numerical solver. Since Model Exchange relies on a single solver, single resolution simulation, it does not exhibit numerical errors induced by coupling and interpolation. In addition, there may be cases under which the simulation runs faster than multi-solver simulation. Other advantages of Model Exchange include the small model size and that no installation of the native platform or presence of the native solver (either in the computer or embedded within the model) is necessary for the model to run. On the other hand, the solver and resolution of the integrated simulation must be compatible with the dynamic characteristics and stiffness of the model, and this can limit the range of applications of this integration method.

FMI for standalone co-simulation is a multi-solver, multi-resolution interface where the model and the native solver are exported into one FMI compliant file. The FMU is imported to a third-party environment where the FMU is ran as a local simulation using its contained native solver under the coordination of a third-party global simulation. The main advantages of this interface over the FMI CS with platform coupling is the lack of network latency-induced communication delays between the two simulations which makes it inherently faster, and the capability of the model to be simulated without the need of an installation of the original platform or the original solver on the host computer. Its main drawback is the considerably large size of the model files which complicates file sharing and storage.

In the FMI co-simulation with platform coupling method, an FMI wrapper is exported rather than the model itself. The FMI wrapper is then imported into a model within a third-party environment and it allows the two environments to establish communication via a virtual server. The two models are simulated in their native environments and the execution of the local simulations and the communication between the models are coordinated by the global simulation running in the third-party environment. This is a multi-solver, multi-resolution method. The advantage of this method over the CS standalone is the considerably smaller model file size. The disadvantages are the slower simulation speed – especially in models with a low computational cost – due to network latency, and the need for a full installation of the original platform.

Models in MATLAB S-Function form make use of the original numerical solver. It is readily importable to MATLAB Simulink and by an array of real-time computers. It is characterized by a high computational efficiency and a small size of model files.

Proprietary target software interfaces that rely on platform coupling are in common use by automotive CAE software. Under this multi-solver, multi-resolution method, each model runs on its native environment. The simulation environments are connected via a virtual server and the execution and communication of the local simulations are coordinated by the global simulation located in one of the environments. The most commonly encountered target software is MATLAB/Simulink. Harnesses that connect tools made by the same company are also popular. CAE tools usually incorporate one interface harness for global and one for local co-simulation mode with a particular platform. The main advantages of this method over all FMI variants and the S-Function is the fact that it is the simplest method to setup and integrate two heterogenous models, and the ability to modify one or more models and rerun the simulation without the need to recompile and reload the models manually, a very desirable feature during the model development phase. The disadvantages of this method are its inherently slower speed due to the latency of the virtual communication network, the lack of versatility, and the requirement of full installations of all the associated software tools.

Independent Co-Simulation Platform or ICOS is developed by VIRTUAL VEHICLE with the purpose of giving the user co-simulation capabilities for a wide array of automotive CAE tool combinations. It is a multi-solver, multi-resolution interface which couples the two environments via a virtual server, and the global simulation on one of the involved environments controls the execution of the local models and the exchange of data. There is one dedicated ICOS variant for each software combination supported by ICOS, and like the proprietary target software interface discussed above, each ICOS variation supports only the specific software combination and hierarchy. The main advantages of ICOS are that it is simple to setup and run, that it supports white box model structure as it allows for the user to make changes to the associated models and directly run the co-simulation without the need to recompile and reload the models manually (a highly desirable feature when developing the model), its capability of connecting to real-time systems, and being supported by AVL Model.CONNECT which is a popular model integration and simulation environment within the automotive industry [25]. Since ICOS relies on platform coupling, simulation can be slower than model exchange or co-simulation standalone options in non real-time systems.

Case Studies

To obtain a first-hand experience on the differences in the behavior between the investigated model integration interfaces, a set of case studies will be presented in the current section. Each case study has been designed to compare different aspects of interface performance. To be consistent with the architecture of the original models used in each case study, the models must be created by the same CAE tool, and that CAE tool must support all interfaces under investigation. Following a research on the offerings of automotive CAE tools, GT-SUITE from Gamma Technologies was found to satisfy the necessary criteria for this comparison as it features a proprietary GT-Simulink platform coupling harness, and model export capability to MATLAB S-Function, FMU CS v1.0, FMU CS v2.0, FMU CS Standalone v1.0, and FMU CS Standalone v2.0. While export to FMU ME is not supported, a single solver, single resolution configuration such as FMI model exchange is less relevant for multi-disciplinary/holistic modelling than the multi-solver options due to the comparatively small range of applications the ME configuration can work under. In addition, a single-solver integrated model does not exhibit coupling and extrapolation errors or communication related delays and the resulting simulation does not differ from other single solver simulations. For these reasons, the absence of FMI ME from the case studies does not alter the weight of this study significantly.

On the Simulink side of the proprietary platform coupling interface, the harness allows for enabling or disabling Direct Feedthrough and for setting the coupling time step which was also set to 1ms. Simulation under both options are ran.

In the case of the S-Function and FMI interfaces, no feedthrough setup option is available. The arrays of the input and the output signals to the model interfaces were compared in terms of numerical values and time step alignment.

The computer hardware composed of an Intel i3-4460 CPU at 3.20GHz, 16GB RAM. The operating system was Windows 7 Enterprise. Connected modelling environments were MATLAB R2018b and GT-ISE v2018.

Case Study 1: Simple Local Model – Open Loop Co-Model, No States

The purpose of this case study is to observe how each integration method affects computation speed and quality of data of an open loop system without states. The simplicity of the model in this case study allows for the easy observation of the differences in the behaviour of each integration method. The GT model consists of a "Gain" element with a unity value. The global model is located in Simulink and consists of a chirp signal source block with an initial frequency of 0.1Hz and a target frequency of 12Hz at the target time of 100 seconds connected to the GT-SUITE-generated local model. The global model uses a fixed timestep solver with a time step size of 1ms and the co-simulation communication time-step is also 1ms.

The wall clock time required for 100 seconds of simulation is shown in the bar chart of figure 2. The MATLAB S-function, FMU v1.0 CS Standalone, and FMU 2.0 CS Standalone are observed to have comparable wall clock times which are considerably shorter than the other options and several times shorter than the simulation duration of 100 seconds (faster than real-time). The proprietary GT – Simulink coupling harness (with DF and without DF), FMU v1.0 CS, and FMU 2.0 CS simulation wall clock times were almost twice the simulation end time (slower than real-time). This is attributed to the latency in the data communication between the coupled platforms.

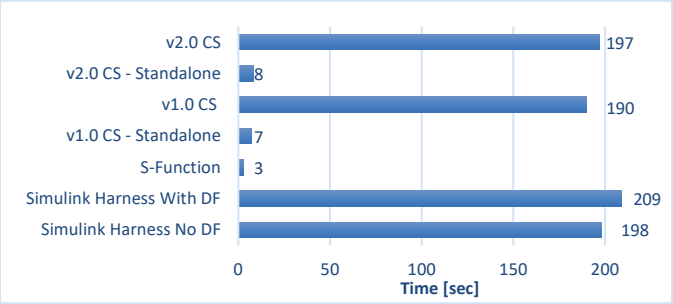

Figure 2. Wall clock time for 100 seconds of simulation of Case Study 1

The values of the model outputs of all integration methods were observed to be identical, but not all interface output arrays align with the one another. In terms of time-step alignment of outputs, for all interfaces but the proprietary GT-Simulink platform coupling with Direct Feedthrough, the output arrays were observed to lag one time step behind the input. The output array of GT-Simulink coupling harness with Direct Feedthrough is in synchronization with the block input.

Case 2: Simple Local Model #2 – Open loop Co-Model with Two States

The purpose of this case study is to observe the effect different model integration methods have on the simulation speed and numerical output of a more complicated model than of case study 1 that has one or more states, yet simple enough for the values of the states to be easily inspected at any given time-step. A simple Mass-Spring-Damper model has been chosen in this role due to it being a well understood system by the majority of the target readers. The local model of the Mass-Spring-Damper built in GT is connected to the global Simulink model consisting of a chirp signal source block with an initial frequency of 0.1Hz and a target frequency of 12Hz at the target time of 100 seconds. The chirp input to the Mass-Spring-Damper model is connected to a force source acting on the mass. The co-models with the seven integration options are simulated for 100 seconds. As the frequency increases, the system briefly goes in and out of resonance. The global model uses a fixed timestep solver with a time step size of 1ms and the co-simulation communication time-step is also 1ms.

The wall clock time for each integration method to complete 100 seconds of simulation of Case Study 2 co-model is shown in the bar chart of figure 3. It is observed that under this case study, the MATLAB S-function, FMU v1.0 CS Standalone, and FMU 2.0 CS Standalone have comparable wall clock simulation times which are considerably shorter than the proprietary GT – Simulink platform coupling (with and without DF), FMU v1.0 CS, and FMU 2.0 CS options, whose wall clock times are approximately two and three times longer than the simulation length in the cases of the proprietary platform coupling interface and the FMU CS respectively. The slower simulation of the FMU CS and the proprietary GT-Simulink harness is attributed to the latency in the data communication between the coupled platforms. While the simulation wall clock time for the MATLAB S-function, and FMU CS Standalone for Case 2 has tripled compared to Case Study 1, thus reflecting the considerable increase in computational load, for the proprietary GT – Simulink platform coupling and FMU CS options, the simulation duration has only increased by approximately 50%, an observation that indicates that communication latency comprises the bulk of the simulation time for Case Studies 1 and 2.

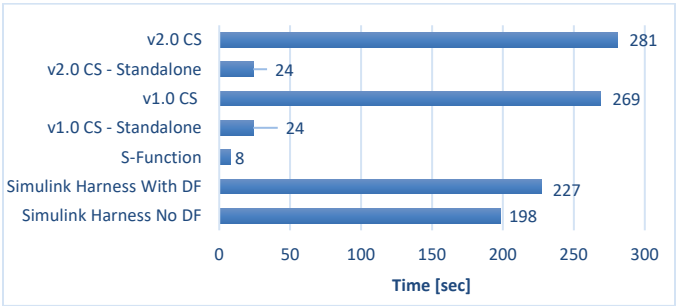

Figure 3. Wall clock time for 100 seconds of simulation of Case Study 2

In terms of the numerical results of Case Study 2, all tested integration methods were observed to produce identical results, and in all tested options, acceleration was observed to not align with force as for the first non-zero value of force, acceleration at the same time step is 0. The output arrays lag two time steps behind the input force

array for all but the proprietary GT-Simulink coupling with Direct Feedthrough interface which lags one time step behind the input force array. Normally, force and acceleration should precede velocity by one time step and velocity should precede displacement by one time step but for all tested interfaces, acceleration, velocity, and displacement appear on the same time step (macro-step).

Case Study 3: Sophisticated Local Model #1 – Open Loop Co-Model with Multiple States

Case Study 3 compares the simulation speed and numerical results of the investigated model integration interfaces with a complicated, computationally intensive multi-state local model. A 1D GT model of a cooling system of an automotive engine serves in the role of the local model with cooling fan speed as the input, and coolant temperature at the engine block inlet temperature as the output. On the global model (Simulink) side, a step function fan speed signal is supplied to the local model with a final value of 1000rpm. The simulation duration is 100 seconds. A fixed step auto numerical solver setting with 1ms time step is set on the Simulink side. In all tests, coupling time step is 1ms.

From the bar chart of wall clock time to complete 100 simulation seconds shown in the of figure 4, it is observed that all tested interfaces exhibit very similar simulation speeds, with wall clock time being approximately 50 times the simulation duration. Considering all other test cases, the computational load of this model outclasses the effect of communication latency as the driving force of simulation speed, and as a result, the relative difference in simulation speed between integration methods is diminished.

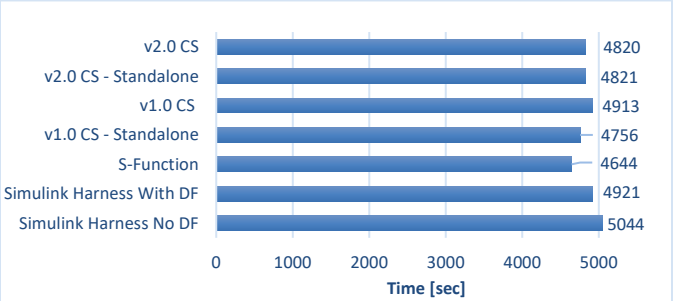

Figure 4. Wall clock time for 100 seconds of simulation of Case Study 3

In terms of the alignment of numerical results of the co-simulation of case study 3, all integration methods produce identical numerical results for most of the simulation interval of 100 seconds. The output arrays are in alignment with the input under all interfaces (measured state equals initial value at t=0 sec) but the proprietary GT-Simulink coupling interface with Direct Feedthrough, appears to start one macro time step before the global model (at macro-step t=0, the measured state is less than the initial state of 300°K), thus, under such setting, the first cell of the output array of the Simulink integration does not have the initial, but rather the value encountered in the second iteration of the other integration methods. The output values of the FMU interfaces for t=0 is 0°K instead of 300°K. For all other macro-steps, the output values of the FMU are identical and in synchronization to those of the S-Function and the GT-Simulink coupling interface without Direct Feedthrough.

Case Study 4: Sophisticated Local Model #1 – Closed Loop Co-Model with Multiple States

Case study 4 investigates the effects of closed loop control on the alignment and consistency of the numerical results. For this purpose, the 1D GT model of a cooling system of an automotive engine from Case Study 3 is used in the role of the local model connected to the global Simulink model of a simple on/off Stateflow controller. Based on the engine block coolant temperature, the controller turns the cooling fan on and off to keep the coolant temperature within the temperature band of 367°K to 380°K. The simulation starts with the controller with the state "off" active. A 'fixed step auto' numerical solver setting with 1ms time step was set on the Simulink side. In all tests, coupling time step was 1ms. Simulation duration is 600 seconds. Since simulation speed for this local model is discussed in Case Study 3, it is not discussed under this case study.

In terms of the output of the local model, prior to first state transition to "on", calculated results are identical to those observed in Case Study 3. Once a state transition is triggered in the on/off type Stateflow controller, the calculated temperatures begin to deviate from one another. Results of FMU v1.0 CS and v2.0 CS are identical. Results of FMU v1.0 CS Standalone and v2.0 CS Standalone are identical. Results of FMU v1.0 CS Standalone and differ from the results of FMU v1.0 CS. Version of FMU does not affect the numerical values.

Since a discrete state controller is used, the easiest way to compare the temporal alignment of the numerical results is to compare the macro-step at which each controller state transition takes place tabulated for all tested integration methods in table 1. State transitions for GT-Simulink coupling interface with Direct Feedthrough enabled, GT-Simulink coupling interface with Direct Feedthrough disabled, MATLAB S-Function, FMU CS, and FMU CS Standalone are observed to occur at different macro-steps. The maximum deviation from all calculated average transition times was -1.441 seconds. Such a deviation may be insignificant for the cooling system application but can potentially be significant in sensitive control applications. Version of FMU does not affect transition times.

Table 1. State transition times of the closed loop controller

Time in Seconds The Stateflow Controller Changes State								
	Platform Coupling		S-function	FMI				
	No DF	DF	-	v1.0 CS Standalone	v1.0 CS	v2.0 CS Standalone	v2.0 CS	Average
on	152.518	152.517	152.616	154.465	152.568	154.465	152.568	153.102
off	205.895	205.892	205.999	207.995	205.948	207.995	205.948	206.525
on	255.371	255.337	255.512	257.393	255.421	257.393	255.421	255.978
off	312.35	312.311	312.5	314.408	312.397	314.408	312.397	312.967
on	360.5	360.406	360.635	362.514	360.574	362.514	360.574	361.102
off	418.755	418.651	418.883	420.771	418.824	420.771	418.824	419.354
on	466.446	466.36	466.535	468.454	466.513	468.454	466.513	467.039
off	525.263	525.18	525.352	527.277	525.331	527.277	525.331	525.859
on	572.739	572.661	572.789	574.654	572.797	574.654	572.797	573.299

The final temperature values of the closed loop co-simulation are tabulated in table 2, and it is observed that GT-Simulink coupling interface with Direct Feedthrough enabled, GT-Simulink coupling interface with Direct Feedthrough disabled, MATLAB S-Function, FMU CS, and FMU CS Standalone all exhibit different final temperature values. Maximum deviation from the average final temperature is 0.137°K which is not of importance for the engine cooling system modelling but can be of higher significance for some sensitive applications.

Table 2. Final temperature values of the closed-loop co-simulation

Time in Seconds When the Stateflow Controller Changes State								
	Platform Coupling		S-function	FMI				
	No DF	DF	-	v1.0 CS Standalone	v1.0 CS	v2.0 CS Standalone	v2.0 CS	Average
Temp. °K	371.7559	371.7392	371.766	372.167	371.7689	372.167	371.7689	371.8761

Since the open-loop co-simulation numerical results of case study 3 under all integration options were observed to be identical, the differences between state transition times as well as between calculated values of the tested integration methods in the closed loop co-simulation of case study 4 are attributed to the method each interface uses to extrapolate the input values to the local models for the micro-steps located between two communication instances.

Case Study 5: Sophisticated Local Model #2 – Open Loop Co-Model with Multiple States

Case Study 5 investigates how each integration method affects computation speed with a numerical model purpose-built to be simulated at a close to real-time rate. For this purpose, a GT-Power crank angle resolved Fast Running Model of a 3.14L Turbo-Diesel internal combustion engine with VGT EGR serves in the role of the local model with pedal position as input, and brake torque, turbo speed, and exhaust temperature as outputs. A fixed step auto numerical solver setting with 1ms time step is set on the Simulink side. In all tests, coupling time step is 1ms. Simulation duration is 100 seconds. The co-simulations are carried out under an engine speed of 2000rpm and a constant pedal position input of 20%. Initial turbo speed is 215000 rpm and initial exhaust temperature value is 700°K.

The wall clock time of 100 seconds of co-simulation for the tested interfaces is shown in the bar chart of figure 5. The S-function, FMU 1.0 CS Standalone and FMU 2.0 CS Standalone are the fastest interfaces as they are more than capable for the given engine speed to allow for the local model to operate as intended as a real-time capable model (and even being faster than real-time). Co-simulations using FMU 1.0 CS, FMU 2.0 CS, and the proprietary GT-Simulink coupling interfaces are 2 to 3 times slower than real-time which defies the purpose of the real-time capable local model.

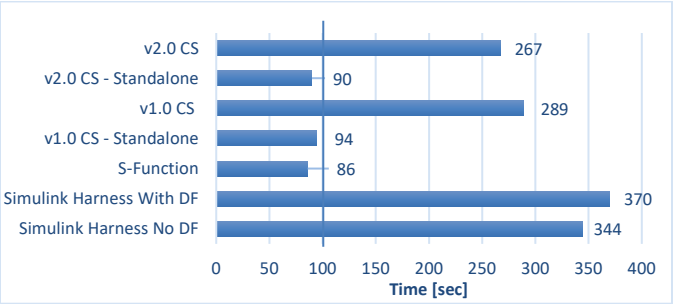

Figure 5. Wall clock time for 100 seconds of simulation of Case Study 5

In terms of the alignment and consistency of numerical results, it is observed that the GT-Simulink coupling interface with Direct Feedthrough disabled, the S-function, and all FMI variants produce almost identical results (the only difference being, that at 0 ms, all FMU outputs are 0) and in complete synchronization from one another. Brake torque for t=0 is 0 Nm, and calculated values become available to the global model at the first macro-step at t=1ms. The initial turbo speed value of 215000 rpm is displayed for t=0 ms, and the first updated value is available at the first macro-step t=1ms. The initial exhaust temperature value of 700°K is carried for two macro-steps unchanged before an updated value is displayed at the third time-step (t=3ms), and the above are observed for all tested interfaces with the exception of the GT-Simulink coupling interface with Direct Feedthrough enabled, under which, all output arrays are offset minus one time step with respect to the results of all other methods, hence the initial turbo speed value is not visible in the results of this method, brake torque update is available at t=0ms, and the first updated value of exhaust temperature is available at the second macro-step (t=2ms).

Engine speed and its influence on simulation speed

For some numerical models, a numerical iteration takes place for every integer multiple of a minimum displacement increment (e.g. for every degree of a rotating shaft). As a result, the time step of such simulation is inversely proportional to the derivative of displacement (angular velocity in this case) and proportional the number of computations per unit of time. This means that for the example crank-angle resolved engine model, the higher the crank angular velocity is, the longer the wall clock time will be required for a given simulation duration. The plot in Figure 6 shows how engine speed affects the simulation real-time ratio of the GT-Power crank angle resolved Fast Running Model of a 3.14L Turbo-Diesel internal combustion engine model simulated in the same i3 3.2GHz 16GB RAM computer. The model is exported to S-function and imported to Simulink where the co-simulation takes place. It is observed that the engine speed influences the simulation speed considerably. At 700 rpm, real-time ratio is less than 0.7. At around 1000 rpm, real-time ratio increases rapidly to 0.9, after which, the rate of increase with speed is less pronounced. After approximately 2130 rpm, the co-simulation becomes slower than real-time. The above highlights the need for a verification of the real-time capability of the co-simulation across the intended speed range for given computer hardware specifications, and the importance of using computers that are capable of handling the real-time co-simulation across the operating envelope of the model.

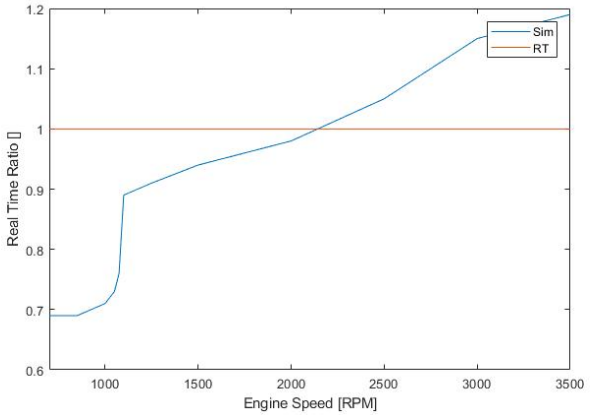

Figure 6. Real-time ratio of Crank-Angle resolved engine model vs. engine speed

Engine speed and signal definition

Since the co-simulation communication time step between each model is constant, the shape of output signals from the local models can be distorted considerably. The distortion is more pronounced for high-frequency undulating signals. The above, combined with the discrete-time signal type of local model outputs can reduce signal quality considerable. As shown in the brake torque vs time plots of figure 7, the instantaneous brake torque curve for the lowest speed of 700 rpm is well defined despite featuring a stepped contour due to the large number of communication steps per cycle. As engine speed increases to 2000 rpm, the frequency of the undulating signal increases and its period decreases, leaving a smaller number of steps to define the curve. At 3000 rpm, the definition of the torque curve is the worst of the three.

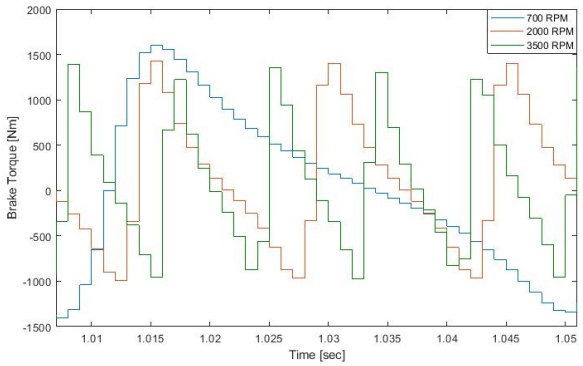

Figure 7. Instantaneous brake torque traces as observed by the global model

Communication time step and signal definition

The instantaneous brake torque output of the same crank angle resolved engine model at 2000 rpm into Simulink using a GT-Simulink coupling interface are shown for Direct Feedthrough enabled and disabled under a 1ms and a 10µs communication time steps/macro-steps in figure 8. It is observed that for the 10µs communication step, the torque curves are very well defined and while they are still stepped, the steps are fine enough for the curve to accurately resemble the real torque curve. In addition, the temporal and shape differences between the Direct Feedthrough enabled and disabled are considerably smaller in the 10µs than in the 1ms co-simulations as the enabled and disabled Direct Feedthrough have a temporal difference of the size of one communication time step and the shapes of the curves approaches continuous time with smaller communication time steps.

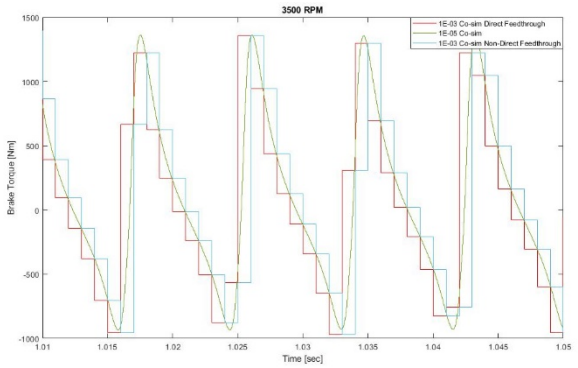

Figure 8. Instantaneous brake torque traces as observed by the global model small and large communication time-steps

To observe the effects of communication step/macro-step size on the integral of the instantaneous torque signals, the curves of torque integral (angular impulse) for a simulation duration of 2 seconds vs. engine speed for the GT-Simulink coupling interface with Direct Feedthrough enabled and disabled for 1ms and 10µs communication time-steps/macro-steps are plotted in figure 9. It is observed that for a given macro-step, the curves for the enabled and disabled Direct Feedthrough are located very close from one another while all 1ms curves are located above the 10µs curves throughout the simulated engine speed range.

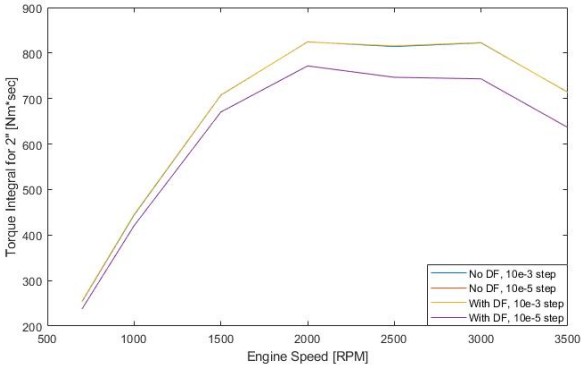

Figure 9. 2" Angular Impulse vs. Engine Speed

While figure 9 shows how a larger communication time-step leads to an overestimation of angular impulse (and therefore mechanical work) by the Simulink global model, if it is assumed that the angular impulse curve under the 10µs macro-step has a negligible error, then the relative error of the 1ms angular impulse curves is calculated with respect to the 10µs curves. The curves of relative error introduced by the 1ms communication time-step co-simulation with and without direct feedthrough enabled are plotted in figure 10. The relative difference between direct feedthrough and non-direct feedthrough curves of the same co-simulation time step is very small (less than 1% under all tested conditions). Calculated angular impulse error exceeds 12% for speeds greater than 3500 rpm. As a result, engine mechanical output into the Simulink model is grossly overestimated and this has the potential for grossly underestimated emissions if instantaneous torque combined with a coarse time step are used.

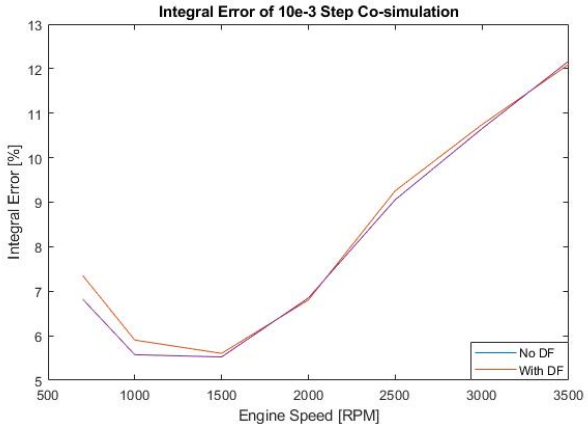

Figure 10. Angular impulse percent error vs. engine speed

Summary

A holistic philosophy of design allows for designing automotive subsystems as parts of the whole system, and as a result, better and more robust vehicle designs reach the final physical testing stage while development time and costs are reduced. Multi-disciplinary simulation is the driver of holistic philosophy of design. The current paper reviewed applications of model integration within the automotive industry and compared the existing model integration interfaces available for multi-disciplinary simulation with respect to several different criteria. The comparison involved carrying out simulations under five different test cases with different combinations of models in order to investigate the workflow and behavior of each interface under different model architectures and computational loads. To avoid inconsistency in practice, the same two modelling environments were used for building co-models in all test cases. The selected environment on which the local/exported models are built is GT-SUITE as it supports a wide variety of interfaces, and although it does not support the FMU Model Exchange (ME) interface, its capability of testing four (seven if interface variations are taken into account) different co-simulation interfaces is of greater importance than the shortcoming of not testing FMU ME which is an interface that is of simpler nature and of a limited application compared to multi-solver methods. A GT-Simulink coupling interface, FMI CS v1.0 and v2.0 are platform coupling methods and require full installation of the local environment (GT-SUITE), while Standalone (solver embedded in model file) interfaces such as MATLAB S-function, FMI CS v1.0 Standalone and v2.0 Standalone do not require a full installation of the local environment but rather, they only require a license of the local environment or solver (GT-SUITE). MATLAB S-function and FMI models can be shared as black box models. The proprietary GT-Simulink coupling interface allows the architecture of the GT model to be accessible and modifiable while integrated in Simulink. The Simulink harness and MATLAB S-function require the use of Bus Creator & Selector or Mux and Demux Simulink objects. Individual signals are not visible on the interface block and this may lead to confusion and slow down integration. In the cases of the FMI, all channel ports are visible on the interface block, thus making the integration more straightforward. The general characteristics of model integration interfaces discussed above are presented in Table 3. Interfaces of the standalone category have been observed to run faster than those of the platform coupling category for all case studies, with the relative difference for simple models being very substantial, but as model complexity increases, the relative difference in speed between standalone and platform coupling methods is gradually diminished. The co-simulation of real-time capable model under the standalone interfaces has been found to be faster than real-time, while under the platform coupling type interfaces, co-simulation is 2 to 3 times slower than real-time. Under open loop simulations, all interfaces generate identical numerical results, but there is a discrepancy in the timing of the outputs as the GT-Simulink coupling interface with Direct Feedthrough generates outputs shifted one time-step earlier relative to the outputs of all other tested interfaces. The first time-step outputs of both FMU CS and CS Standalone types is 0. For local models whose simulation time-step is dependent on the speed of the simulated moving parts, such as the crank-angle resolved engine models, the definition of the engine model outputs into the global model deteriorates and the wall clock simulation time for a given simulation duration increases, as engine speed increases. After a certain engine speed threshold, the co-simulation becomes slower than real-time, a behavior that highlights the need for verifying the model is real-time capable under the complete operating envelope when simulated on a computer with a given set of hardware specifications. Co-simulation communication has been observed to reduce the resolution of local model output signals, especially of undulating, high-frequency form. The larger the co-simulation communication time-step (macro-step) is, the greater the distortion of the signal. The angular impulse calculated from the instantaneous torque of the tested crank angle-resolved engine model was found to be overestimated with a relative error of up to 12% for higher engine speeds under a 1ms macro-step.

Table 3. General characteristics of model integration interfaces

	Model Integration Interfaces					
Criteria	FMU ME	MATLAB S-function	FMU CS Standalone	FMU CS Platform Coupling	ICOS	Proprietary Platform Coupling Harness
Support by software	Very High	Very High	Very High	Very High	High (Virtual Vehicle)	High
Simulation Speed	Very High	Very High	Very High	Ranges from Slow to High	Ranges from Slow to High	Ranges from Slow to High
Model Configurability	Poor	Poor	Poor	Poor	Excellent	Excellent
Simplicity in Procedure Setup	Simple	Simple	Simple	Simple	Very Simple	Very Simple
Multiple solvers	No	Yes	Yes	Yes	Yes	Yes
Weak coupling	No					
Installation of original platform	No	No	No	Yes	Yes	Yes
Model Access	Black Box	Black Box	Black Box	Black Box	White Box	White Box
Model file size	Small	Small	Very Large	Very Large	Small	Small

Conclusions

From the above, it is concluded that there is no all-around best model integration method, but rather, each integration solution shares a different set of advantages and disadvantages which may make it more suitable for a particular application or developmental stage, and for this reason, using different interfaces in different applications and developmental stages can benefit the development process. For the same reasons, the selection and configuration of a model integration interface must be the product of careful consideration of the nature of the interface and the application. The current document presented a guideline to point engineers towards a certain integration direction for a given set of requirements.

The model integration interfaces discussed above have different degrees of support by software companies, with the FMI standard

being the most popular since it is supported by most automotive CAE tools and a number of real-time computers. FMI ME is best suited in cases when the model is shared in black box form and the solver of the third-party simulation can handle the dynamics of the imported FMU model. FMU CS Standalone is suitable for cases where the model is shared in black box form, a fast-running multi-solver simulation is necessary, and the end-user of the model does not have an installation of the original platform, but a solver license is available. Export to an S-Function is a good choice when the model is shared in black box form, the target platform is Simulink, and a high simulation speed is required, or when the target platform is a real-time computer. Platform coupling using a proprietary interface is best suited in cases when frequent changes in the structure of the connected models must be made followed by simulations to observe the effects of the changes e.g. during model development.

Co-simulation induced errors can affect calculated fuel consumption, emissions, NVH, as well as high-frequency sensor models such as knock sensors etc. For some studies in which pulsating values are not necessary for the study, and a coarse co-simulation time-step is preferred due to inherently higher co-simulation speeds, using cycle-averaged values as local model outputs can improve the accuracy of the results. Models of this type are often referred to as 'mean value' and their suitability for vehicle modelling applications has been discussed [26]. Under closed loop simulations, differences are observed in numerical outputs of the test cases. For the closed-loop controlled engine cooling system, the numerical outputs were identical up until the first state transition. After the first state transition, numerical results and state transitions timings started to diverge between the tested interfaces. The observed differences are considered acceptable for the nature of the test case example but could be unacceptable for more sensitive applications.

In closing it is relevant to state that the majority of phenomenon highlighted here apply in general to the numerical modelling of any system combination that requires coupling of physical systems with widely differing magnitudes of temporal derivates or frequencies. In this context co-simulation presents additional difficulties by forcing data exchange interfaces with related system coupling errors that may not be desirable for solution accuracy and/or computational efficiency.

Contact Information

Dr. Nikolaos Kalantzis
Dept. of Aeronautical and Automotive Engineering
Loughborough University
Loughborough
LE11 3TU
n.kalantzis@lboro.ac.uk

Acknowledgments

The authors would like to acknowledge the funding support from the Innovate UK and the Advanced Propulsion Centre (APC) for carrying out this work. We would also like to acknowledge the support from Jaguar Land Rover Automotive PLC for providing resources and the support of Gamma Technologies for providing GTSUITE software and associated support.

Definitions/Abbreviations

ABS	anti-lock braking system
CAE	computer aided engineering
CAN	controller area network
DC	direct current
EiL	engine-in-the-loop
ESP	electronic stability program
FMI	functional mock-up interface
FMU	functional mock-up unit
FPGA	field-programmable gate array
GTDI	gasoline turbocharged direct injection
HiL	hardware-in-the-loop
ICE	internal combustion engine
MiL	model-in-the-loop
NVH	noise, vibration, and harshness
PHEV	parallel hybrid electric vehicle

References

1. P. Le Marrec *et al.*, "Hardware , Software and Mechanical Cosimulation for Automotive Applications," in *Ninth International Workshop on Rapid System Prototyping (Cat. No.98TB100237)*, 1998.
2. A. Karvonen and T. Thiringer, "Co-Simulation and Harmonic Analysis of a Hybrid Vehicle Traction System," *2015 IEEE Veh. Power Propuls. Conf. VPPC 2015 - Proc.*, 2015.
3. W. Chen, M. Klomp, and S. Ran, "Real-time Co-simulation Method Study for Vehicle Steering and Chassis System," *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 273–278, 2018.
4. L. Mikelsons and R. Samlaus, "Towards Virtual Validation of ECU Software using FMI," in *Proceedings of the 12th International Modelica Conference May 15-17, 2017,* 2017, pp. 307–311.
5. S. Klein *et al.*, "Engine in the Loop: Closed Loop Test Bench Control with Real-Time Simulation," *SAE Int. J. Commer. Veh.*, vol. 10, no. 1, pp. 2017-01–0219, 2017.
6. I. M. Khan, M. Datar, W. Sun, G. Festag, T. Bin Juang, and N. Remisoski, "Multibody Dynamics Cosimulation for Vehicle NVH Response Predictions," *SAE Int. J. Veh. Dyn. Stability, NVH*, vol. 1, no. 2, pp. 131–136, 2017.
7. N. Pedersen *et al.*, "Co-simulation of distributed engine control system and network model using FMI & SCNSL," *IFAC-PapersOnLine*, vol. 48, no. 16, pp. 261–266, 2015.
8. T. Wu, K. Han, L. Pei, and C. Zhao, "Co-Simulation Study of Coordinated Engine Control Focusing on Tracked Vehicle Shift Quality," *J. Autom. Control Eng.*, vol. 2, no. 2, pp. 160–165, 2014.
9. J. Fitzgerald *et al.*, "Collaborative Modelling and Co-simulation in the Development of Dependable Embedded Systems," in *Integrated Formal Methods - IFM*, 2010, pp. 12–26.
10. M. Aslan, H. Oğuztüzün, U. Durak, and K. Taylan, "MOKA : An Object-Oriented Framework for FMI," *47th Summer Comput. Simul. Conf. 2015*, no. SummerSim '15, pp. 1–8, 2015.
11. F. R. L.M. Reyneri, E. Bellei, E. Bussolino, L. Mari, "Codesign and Cosimulation of Automotive Systems Based on Matlab /

Simulink," in *Seminario Anual de Automática, Electrónica Industrial e Instrumentación, SAAEI*, 2002, no. May.
12. L. T. Kyllingstad, M. Rindarøy, and D. Eirik, "Distributed Co-Simulation of Maritime Systems and Operations," *CoRR*, vol. 1701.00997, 2017.
13. P. Casoli, A. Gambarotta, N. Pompini, and L. Riccò, "Development and application of co- simulation and ' control - oriented ' modeling in the improvement of performance and energy saving of mobile machinery," *Energy Procedia*, vol. 45, pp. 849–858, 2014.
14. B. Zhang *et al.*, "Component Tests Based on Vehicle Modeling and Virtual Testing," *SAE Tech. Pap. 2017-01-0384*, 2017.
15. G. Li, T. Wang, R. Zhang, F. Gu, and J. Shen, "An Improved Optimal Slip Ratio Prediction considering Tyre Inflation Pressure Changes," *J. Control Sci. Eng.*, vol. 2015, 2015.
16. F. Xie, J. Wang, and Y. Wanga, "Modelling and Co-simulation Based on AMESim and Simulink for Light Passenger Car with Dual State CVT," in *Procedia Engineering (2011) 16 363-368*, 2011, vol. 16, no. Apee, pp. 363–368.
17. M. Maharun, M. Bin Baharom, and M. S. Mohd, "Modelling and control of 4WD parallel split hybrid electric vehicle converted from a conventional vehicle," *World J. Model. Simul.*, vol. 9, no. 1, pp. 47–58, 2013.
18. S. Li, L. Zhao, and C. Yang, "Co-Simulation Study for Fuzzy ESP Control Strategy on Vehicle," *Open Mech. Eng. J.*, pp. 682–688, 2014.
19. O. Özener and L. Allouchery, "ISTANBUL METROBUS LINE FUEL CONSUMPTION OPTIMIZATION VIA 3D ROAD MODEL BY USING AVL CRUISE & IPG TRUCK MAKER CO-SIMULATION," *Int. J. Adv. Automot. Technol.*, vol. 1, no. 2, pp. 100–104, 2017.
20. J. J. Eckert, F. M. Santiciolli, S. Costa, F. C. Corrêa, H. J. Dionísio, and F. G. Dedini, "VEHICLE GEAR SHIFTING CO - SIMULATION TO OPTIMIZE PERFORMANCE AND FUEL CONSUMPTION IN THE BRAZILIAN STANDARD URBAN DRIVING CYCLE," in *XXII Simpósio Internacional de Engenharia Automotiva*, 2014, vol. 1.
21. T. Fletcher, N. Kalantzis, M. Cary, B. Lygoe, A. Pezouvanis, and K. Ebrahimi, "Automated Model Based Engine Calibration Procedure using Co-Simulation," in *4th Biennial International Conference on Powertrain Modelling and Control (PMC 2018)*, 2018, p. 118.
22. C. Zhang and Z. Sun, "Using variable piston trajectory to reduce engine-out emissions," *Appl. Energy*, vol. 170, pp. 403–414, 2016.
23. R. Hallqvist, R. Braun, and P. Krus, "Early Insights on FMI-based Co-Simulation of Aircraft Vehicle Systems," *Proc. 15th Scand. Int. Conf. Fluid Power, 15th Scand. Int. Conf. Fluid Power, Fluid Power Digit. Age, SICFP'17, June 7-9 2017 - Linköping, Sweden*, vol. 144, pp. 262–270, 2017.
24. C. Bertsch, E. Ahle, and U. Schulmeister, "The Functional Mockup Interface - seen from an industrial perspective," *Proc. 10th Int. Model. Conf. March 10-12, 2014, Lund, Sweden*, vol. 96, pp. 27–33, 2014.
25. J. Z. (Volvo) Pamela Innerwinkler (VIF), Georg Stettinger (VIF), Ralph Weissnegger (CISC), Cihangir Derse (TF), Eren Aydemir (FO), "Modular co-simulation architecture plan," 2018.
26. R. P. Osborne and N. Weaver, "Optimum engine models for diesel automotive powertrain development processes," *IPDS 2006: Integrated Powertrain and Driveline Systems 2006*. pp. 67–76, 2006.